The Object-Oriented Thought Process

Chapter 03

Advanced Object-Oriented Concepts

Constructors

Constructors are used to initialize objects.

- In Java and C#, constructors are methods that share the same name as the class.
- Visual Basic .NET uses the designation *New.*
- Objective-C uses the *init keyword.*

When a Constructor is Called

When a new object is created, one of the first things that happens is that the constructor is called.

- The constructor creates a new instance of the class,
  - thus allocating the required memory.
- Then the constructor itself is called, passing the arguments in the parameter list.
  - Providing the opportunity to attend to the appropriate initialization.

What's Inside a Constructor

Perhaps the most important function of a constructor is to initialize the memory allocated.

- In short, code included inside a constructor should set the newly created object to its initial, stable, safe state.

The Default Constructor

If the class provides no explicit constructor, a default constructor will be provided.

- It is important to understand that at least one constructor always exists, regardless of whether you write a constructor yourself.
- If you do not provide a constructor, the system will provide a default constructor for you.

Using Multiple Constructors

In many cases, an object can be constructed in more than one way.

- To accommodate this situation, you need to provide more than one constructor.
- This is called *overloading a method (overloading pertains to all methods, not just constructors).*
  - Most OO languages provide functionality for overloading a method.

Overloading Methods

Overloading allows a programmer to use the same method name over and over.

- As long as the signature of the method is different each time.
- The signature consists of the method name and a parameter list

The Superclass

When using inheritance, you must know how the parent class is constructed.

- Inside the constructor, the constructor of the class's superclass is called.
- Each class attribute of the object is initialized.
- The rest of the code in the constructor executes.

Designing Constructors

It is good practice to initialize all the attributes.

- In some languages, the compiler provides some sort of initialization.
- As always, don't count on the compiler to initialize attributes!
- Constructors are used to ensure that the application is in a stable (or safe) state.

Error Handling

Assuming that your code has the capability to detect and trap an error condition, you can handle the error in several ways:

- Ignore the problem—not a good idea!
- Check for potential problems and abort the program when you find a problem.
- Check for potential problems, catch the mistake, and attempt to fix the problem.
- Throw an exception. (Often this is the preferred way to handle the situation.)

The Concept of Scope

- Multiple objects can be instantiated from a single class.
  - Each of these objects has a unique identity and state.
  - Each object is constructed separately and is allocated its own separate memory.

Types of Scope

- Methods represent the behaviors of an object; the state of the object is represented by attributes.
  - Local attributes
  - Object attributes
  - Class attributes

Local Attributes

Local attributes are owned by a specific method

- Local variables are accessible only inside a specific method.
- In Java, C#, C++ and Objective-C, scope is delineated by curly braces ({ }).

```
public method() {
    int count;
}
```

Object Attributes

In many design situations, an attribute must be shared by several methods within the same object.

```
public class Number {
        int count; // available to both method1 and method2
        public method1() {
                count = 1;
        }
        public method2() {
                count = 2;
        }
}
```

Class Attributes

It is possible for two or more objects to share attributes. In Java, *C#*, C++ and Objective-C, you do this by making the attribute *static:*

```
public class Number {
    static int count;
        public method1() {
    }
}
```

Operator Overloading

Some OO languages allow you to overload an operator.

- C++ is an example of one such language. Operator overloading allows you to change the meaning of an operator.
- More recent OO languages like Java, .NET, and Objective-C do not allow operator overloading.

Multiple Inheritance

Multiple inheritance allows a class to inherit from more than one class.

- Multiple inheritance can significantly increase the complexity of a system,
- Java, .NET, and Objective-C do not support multiple inheritance (C++ does).
- In some ways interfaces compensates for this.

Object Operations

Comparing primitive data types is quite straightforward.

- Copying and comparing objects is not quite as simple.
- The problem with complex data structures and objects is that they might contain references.
  - Simply making a copy of the reference does not copy the data structures or the object that it references.